Optimizers Explained

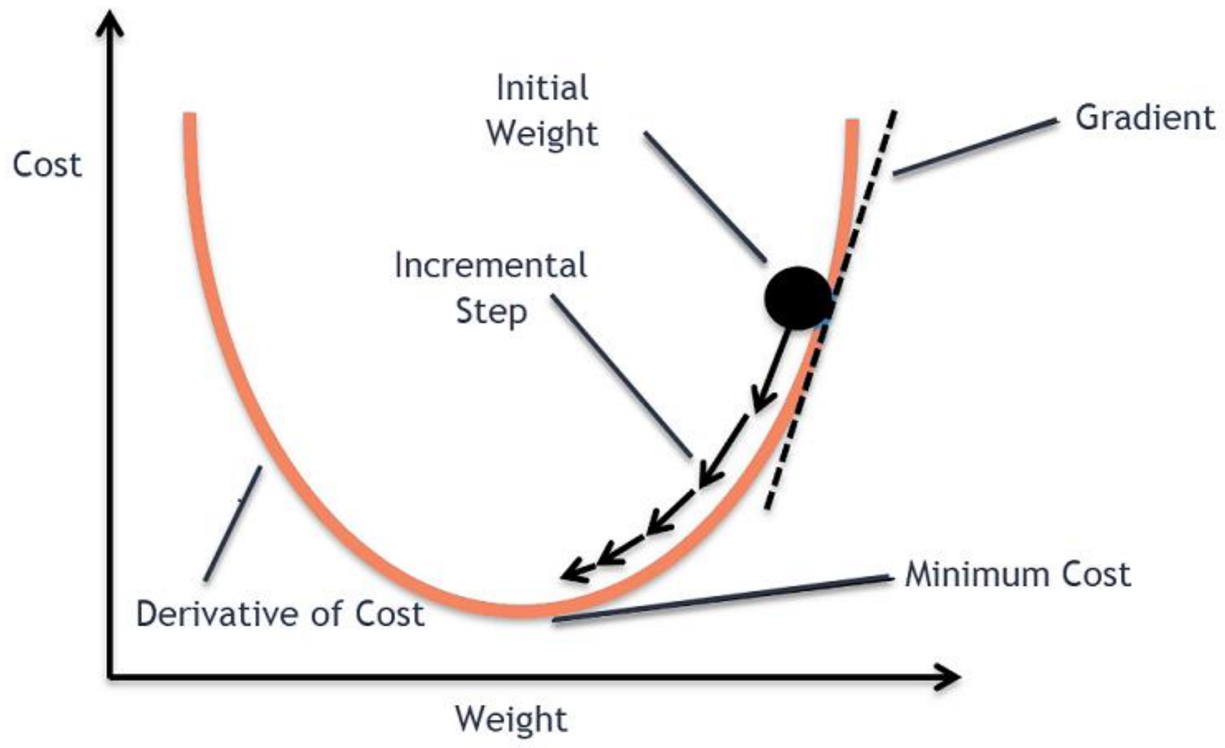
Optimizers are used to optimize the cost function of the model.

Reference: https://www.youtube.com/watch?v=TudQZtgpoHk&t=5192s

Types of optimizers:

1. Gradient Descent
2. Stochastic Gradient Descent (SGD)
3. Mini Batch Gradient Descent
4. SGD with Momentum
5. Adagrad (Adaptive Gradient descent)
6. Adadelta and RMSProp
7. Adam (combination of RMSProp and Momentum)

1. Gradient Descent: (aka Batch Gradient Descent)



- Gradient descent is an optimization algorithm used in machine learning to minimize the cost function by iteratively adjusting parameters in the direction of the negative gradient, aiming to find the optimal set of parameters.

- In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters.

- So that's just one step of gradient descent in one epoch.

- Batch Gradient Descent is great for convex or relatively smooth error manifolds (convex error manifold having one global minima I.e., no local minima).

To achieve this goal, it performs two steps iteratively:

1. **Compute the gradient** (slope), the first order derivative of the function at that point
2. **Make a step (move) in the direction opposite to the gradient**, opposite direction of slope increases from the current point by alpha times the gradient at that point

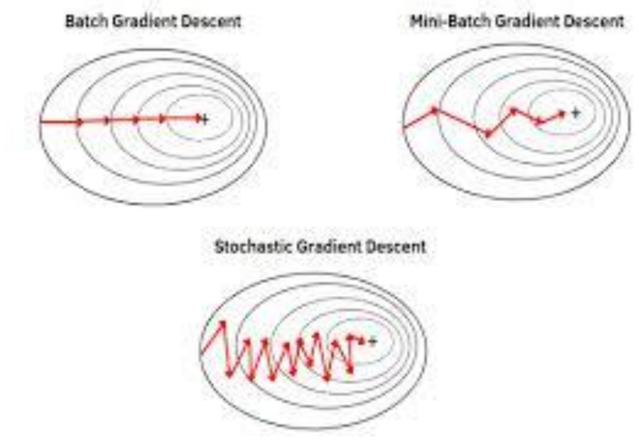
2. Stochastic Gradient Descent:

- In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step.
- SGD can be used for larger datasets. It converges faster when the dataset is large as it causes updates to the parameters more frequently.
- Since we are considering just one example at a time the cost will fluctuate over the training examples and it will **not** necessarily decrease. But in the long run, you will see the cost decreasing with fluctuations.

3. Mini Batch Gradient Descent:

- Neither do we use all the dataset all at once nor we use the single example at a time.
- We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini batch.

Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
• Entire dataset for updation	• Single observation for updation	• Subset of data for updation
• Cost function reduces smoothly	• Lot of variations in cost function	• Smoother cost function as compared to SGD
• Computation cost is very high	• Computation time is more	• Computation time is lesser than SGD
		• Computation cost is lesser than Batch Gradient Descent



4. SGD with Momentum:

SGD with momentum is a method which helps accelerate gradients vectors in the right directions, thus leading to faster converging.

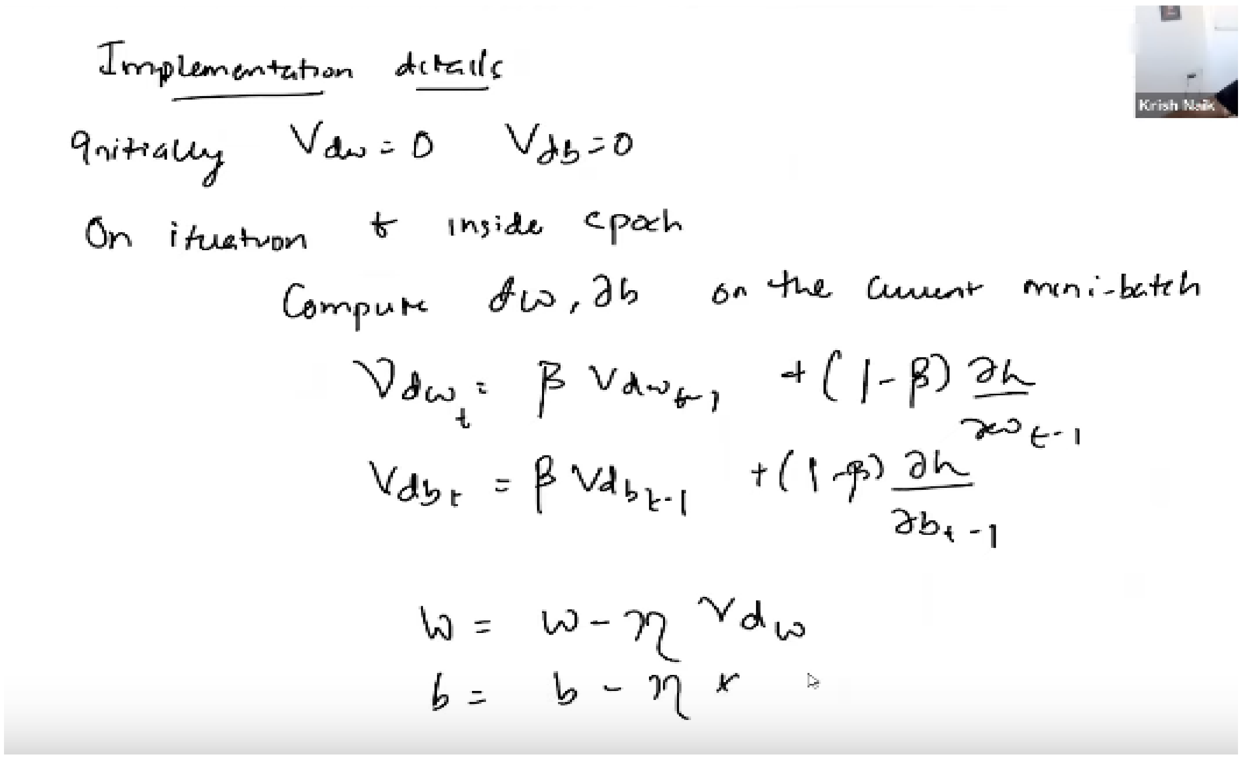
It is based on exponentially weighted moving average (**EWMA**) technique which is generally used to check the trend in time series data.

Momentum accumulates the gradient of the past steps to determine the direction to go.

This helps us move more quickly towards the minima. For this reason, momentum is also referred to as a technique which dampens oscillations in our search.

In practice, the coefficient of momentum is initialized at 0.5, and gradually annealed to 0.9 over multiple epochs.

Refer: https://www.youtube.com/watch?v=TudQZtgpoHk&t=5192s **(from 30 to 56 minutes)**



Implementation details

Initially $V_{dw} = 0$ $V_{db} = 0$

On iteration t inside epoch

Compute $dw, \partial b$ on the current mini-batch

$$V_{dw_t} = \beta V_{dw_{t-1}} + (1-\beta)\frac{\partial L}{\partial w_{t-1}}$$

$$V_{db_t} = \beta V_{db_{t-1}} + (1-\beta)\frac{\partial L}{\partial b_{t-1}}$$

$$w = w - \eta V_{dw}$$

$$b = b - \eta$$

5. Adagrad: (Adaptive Gradient Descent)

- It uses different learning rates for each iteration.
- The change in learning rate depends upon the difference in the parameters during training. The more the parameters get changed, the more minor the learning rate changes.
- This modification is highly beneficial because real-world datasets contain sparse as well as dense features. So, it is unfair to have the same value of learning rate for all the features.
- The Adagrad algorithm uses the formula below to update the weights. Here the alpha(t) denotes the different learning rates at each iteration, n is a constant, and E is a small positive to avoid division by 0.

$$\mathrm{w_t} = \mathrm{w_{t-1}} - \eta'_{\mathrm{t}} \frac{\partial \mathrm{L}}{\partial \mathrm{w(t-1)}} \qquad \eta'_{\mathrm{t}} = \frac{\eta}{\mathrm{sqrt}(\alpha_{\mathrm{t}} + \epsilon)}$$

- The benefit of using Adagrad is that it abolishes the need to modify the learning rate manually. It is more reliable than gradient descent algorithms and their variants, and it reaches convergence at a higher speed.

- One downside of the AdaGrad optimizer is that it decreases the learning rate aggressively and monotonically. There might be a point when the learning rate becomes extremely small. This is because the squared gradients in the denominator keep accumulating, and thus the denominator part keeps on increasing.
- Due to small learning rates, the model eventually becomes unable to acquire more knowledge, and hence the accuracy of the model is compromised.

6. Adadelta and RMSProp:

Refer video: https://www.youtube.com/watch?v=TudQZtgpoHk&t=5192s (1:17 to 1:28 minutes)

7. Adam (combination of RMSProp and Momentum)

Refer video: https://www.youtube.com/watch?v=TudQZtgpoHk&t=5192s (1:28 to end of video)